Université Sultan Moulay Slimane
Ecole Nationale des Sciences Appliquées Khouribga

RAPPORT PROJET BIG DATA

SYSTEME DE RECOMMANDATION DE FILMS

REALISE PAR :

- SOKRI YASSER – N°41
- OUBADOU ABDELLAH – N°34
- YASSINE YOUSSEF – N°42

TABLE DES MATIERES

REMERCIEMENT :

Je tiens tout d'abord à remercier Dieu le tout puissant et miséricordieux, qui m'a donné la force et la patience d'accomplir ce travail.

Mes vifs remerciements s'adressent également à Madame Soussi Nassima, pour sa bienveillance et sa générosité et la grande patience dont il a su faire preuve durant le module d'Introduction au Big Data.

Enfin, je tiens à remercier tous ceux qui ont contribué de près ou de loin à la réalisation de ce modeste projet.

INTRODUCTION

Les systèmes de recommandation sont des outils permettant d'interagir avec des espaces d'information vastes et complexes. Ils offrent une vision personnalisée de ces espaces, en priorisant les éléments susceptibles d'intéresser l'utilisateur. La recherche sur les systèmes de recommandation a intégré une grande variété de techniques d'intelligence artificielle, y compris l'apprentissage automatique, les données l'exploration de données, la modélisation des utilisateurs, le raisonnement basé sur des cas et la satisfaction des contraintes, entre autres. Les recommandations personnalisées sont une partie importante de nombreuses applications de commerce électronique en ligne telles qu'Amazon.com, Netflix et YouTube. Cette riche expérience d'application pratique a inspiré les chercheurs à étendre la portée des systèmes de recommandation dans des domaines nouveaux et difficiles. L'objectif d'un système de recommandation est de générer des recommandations significatives à un ensemble d'utilisateurs pour des articles ou des produits susceptibles de les intéresser. Nous visons ici à faire une étude comparative entre les différents modèles de système de recommandation répandus et à déterminer les avantages et les inconvénients ainsi que les raisons de chacun de ces modèles.

Un système de recommandation peut être défini comme "Tout système qui produit des recommandations individualisées en sortie ou a pour effet de guider l'utilisateur de manière personnalisée vers des objets intéressants ou utiles

dans un large espace d'options possibles".

Les systèmes de recommandation forment un type spécifique de technique de filtrage d'informations qui tente de présenter des éléments d'information (films, musique, livres, images d'actualités, pages Web, etc.) susceptibles d'intéresser l'utilisateur. En règle générale, un système de recommandation compare le profil de l'utilisateur à certaines

caractéristiques de référence et cherche à prédire la "note" qu'un utilisateur donnerait à un élément qu'il n'avait pas encore pris en compte.

Les principales approches des systèmes de recommandation peuvent être globalement classée comme :

- *Filtrage collaboratif (CF)* : dans les systèmes CF, un utilisateur se voit recommander des éléments en fonction des évaluations passées de tous les utilisateurs collectivement.
- *Recommandation basée sur le contenu* : ces approches recommandent des éléments dont le contenu est similaire aux éléments que l'utilisateur a aimés dans le passé, ou qui correspondent aux attributs de l'utilisateur.
- *Approches hybrides* : Ces méthodes combinent à la fois une approche collaborative et une approche basée sur le contenu.

Nous avons pris le jeu de données « movielens » pour ce projet et utilisé la recommandation basée sur le contenu.

JEU DE DONNEES :

Le jeu de données utilisé est un jeu de données movielens avec plus d'un million d'éléments de données

Ce jeu de données décrit l'activité de notation 5 étoiles et de marquage en texte libre de [MovieLens](http://movielens.org), un service de recommandation de films. Il contient 100004 classements et 1296 applications de balises sur 9125 films. Ces données ont été créées par 671 utilisateurs entre le 09 janvier 1995 et le 16 octobre 2016. Cet ensemble de données a été généré le 17 octobre 2016.

Les utilisateurs ont été sélectionnés au hasard pour l'inclusion. Tous les utilisateurs sélectionnés avaient évalué au moins 20 films. Aucune information démographique n'est incluse. Chaque utilisateur est représenté par un identifiant et aucune autre information n'est fournie.

STRUCTURE DE FICHIER DE DONNEES D'APPRECIATION – RATING.CSV

Toutes les notes sont contenues dans le fichier `ratings.csv`. Chaque ligne de ce fichier après la ligne d'en-tête représente une évaluation d'un film par un utilisateur et a le format suivant :

userId, movieId, évaluation

Les lignes de ce fichier sont d'abord classées par userId, puis, au sein de l'utilisateur, par movieId. Les notes sont faites sur une échelle de 5 étoiles, avec des incréments d'une demi-étoile (0,5 étoiles - 5,0 étoiles).

STRUCTURE DU FICHIER DE DONNEES DE FILMS – MOVIES.CSV

Les informations sur les films sont contenues dans le fichier `movies.csv`. Chaque ligne de ce fichier après la ligne d'en-tête représente un film et a le format suivant :

ID film, titre, genres

Les genres sont une liste séparée par des tubes et sont sélectionnés parmi les suivants :

- Action
- Aventure
- Animations
- Enfants
- La comédie
- Crime
- Documentaire
- Drame
- Fantaisie
- Film noir
- Horreur
- Comédie musicale
- Romance
- (aucun genre répertorié)
- ...

FRAMEWORK UTILISE :

➢ Nous avons utilisé Hadoop MapReduce pour le projet

MapReduce est un modèle (ou structure) de programmation disponible dans les environnements Hadoop qui est utilisé pour accéder aux big data stockées dans le Hadoop File System (HDFS). MapReduce est un élément essentiel et fait partie intégrante du fonctionnement de l'environnement Hadoop.

MapReduce facilite les traitements concurrents en divisant les péta-octets de données en volumes plus petits et en les traitant en parallèle sur des serveurs standard dédiés à Hadoop. Pour résumer, MapReduce agrège les données de plusieurs serveurs et renvoie un résultat consolidé à l'application.

Avec MapReduce, plutôt que d'envoyer les données là où se trouve l'application ou les algorithmes, les algorithmes sont exécutés sur le serveur où se trouvent déjà les données, ce qui a pour effet d'accélérer le traitement. L'accès aux données et le stockage des données se font sur disque - les entrées sont généralement stockées sous forme de fichiers contenant des données structurées, semi-structurées ou non structurées, et la sortie est également stockée dans des fichiers.

SYSTEME DE RECOMMANDATION BASE SUR LE CONTENU :

Le système de recommandation basé sur le contenu fonctionne avec les données fournies explicitement par l'utilisateur, c'est-à-dire la notation ou le flux implicite d'un élément à un autre. Sur la base des données obtenues de l'utilisateur, nous créons un profil d'utilisateur. Ce profil d'utilisateur nous aide à suggérer des films. Au fur et à mesure que l'utilisateur met plus de données dans son profil d'utilisateur en donnant la note, la recommandation peut faire de plus en plus de précision.

Le concept de TF-IDF (Term Frequency – Inverse Document Frequency) est utilisé pour déterminer l'importance relative d'un document/film.

Dans le système de recommandation de film, nous avons considéré que la note supérieure à 2,5 sur 5,0 a un film aimé et l'a représenté au format binaire comme 1, et les films qui ont une note

Movies	Adventure	Action	Fantasy	Drama	Crime	Thriller		User-1
Star Wars IV	1	1	1	0	0	0		1
Saving Private Ryan	0	0	0	1	0	0		
American Beauty	0	0	0	1	0	0		
City of Gold	0	0	0	1	1	0		-1
Interstellar	0	0	1	1	0	0		1
The Matrix	1	1	1	0	0	1		

inférieure à 2,5 sont considérés comme des films détestés et reçoivent une valeur de négation du film aimé, c'est-à-dire -1.

D'après l'image ci-dessus, l'utilisateur 1 aimait Star Wars IV et Interstellar et n'aimait pas City of Gold. Par conséquent, le score est donné respectivement. À partir de l'ensemble de données movielens, nous avons compris dans quel genre le film se situe. À partir de ces informations, nous représentons la présence du genre dans un film particulier soit par 0/1.

Movies	Adventure	Action	Fantasy	Drama	Crime	Thriller		Total Attributes		User-1
Star Wars IV	0.57735027	0.57735027	0.57735027	0	0	0		3		1
Saving Private Ryan	0	0	0	1	0	0		1		
American Beauty	0	0	0	1	0	0		1		
City of Gold	0	0	0	0.70710678	0.70710678	0		2		-1
Interstellar	0	0	0.70710678	0.70710678	0	0		2		1
The Matrix	0.5	0.5	0.5	0	0	0.5		4		
...	..	..	..	..	..	..				
User-1 Score										
DF	4	5	6	5	3	7				
IDF	0.39794001	0.30103	0.22184875	0.30103	0.52287875	0.15490196				

Dans l'image ci-dessus, nous avons calculé le score normalisé pour chaque genre dans chaque film. Par exemple, pour le film Interstellar, un nombre total d'attributs est de 2 (Fantastique et Drame).

Nous normalisons maintenant le score du genre à 1/sqrt(2) = 0,70710678. DF (Document Frequency) est calculé en prenant un nombre total d'un type dont le genre particulier est apparu dans l'ensemble de données. À des fins de démonstration, nous avons pris un nombre total de films = 10 dont un certain nombre de films de genre d'aventure sont 4. À partir de ces DF, nous calculons les IDF, c'est-à-dire IDF = log (nombre total de films/DF).

Movies	Adventure	Action	Fantasy	Drama	Crime	Thriller	Total Attributes	User-1	Prediction
Star Wars IV	0.57735027	0.57735027	0.57735027	0	0	0	3	1	0.66332644
Saving Private Ryan	0	0	0	1	0	0	1	-	0
American Beauty	0	0	0	1	0	0	1	-	0
City of Gold	0	0	0	0.70710678	0.70710678	0	2	-1	-0.2614394
Interstellar	0	0	0.70710678	0.70710678	0	0	2	1	0.27992912
The Matrix	0.5	0.5	0.5	0	0	0.5	4	-	0.61318303
...	..	..	..	..	..	..			
User-1 Score	1.07735027	1.07735027	1.78445705	0	-0.7071068	0.5			
DF	4	5	6	5	3	7			
IDF	0.39794001	0.30103	0.22184875	0.30103	0.52287875	0.15490196			

Dans l'image ci-dessus, nous avons trouvé la prédiction pour User-1 pour chaque film de la base de données. Le produit scalaire du vecteur de film et du vecteur IDF nous donne les scores pondérés de chaque film. Ces scores pondérés sont à nouveau utilisés pour un produit scalaire avec le score du profil

utilisateur, cela nous donne le score de prédiction de chaque film dans l'ensemble de données.

Prediction (The Matrix, User-1) =

(0.5*0.39794001*1.07735027) + (0.5*0.30103*1.07735027) + (0.5*0.22184875*1.78445705) + (0.5*0.15490196*0.5)

AVANTAGES DE SYSTEME DE RECOMMANDATION BASE SUR LE CONTENU :

En utilisant Content-Based Recommender, nous pouvons obtenir la recommandation de film uniquement avec le profil de cet utilisateur particulier. Comme l'utilisateur lui-même donne les notes, il est plus probable que l'utilisateur puisse aimer des films de genre similaires.

LIMITATION DE SYSTEME DE RECOMMANDATION BASE SUR LE CONTENU :

Limitation générale de ce que nous avons rencontré est le cas où l'utilisateur n'a pas regardé un genre particulier de genre. Par exemple, dans la dernière image, nous pouvons voir que puisque l'utilisateur n'a regardé aucun type de film dramatique, la prédiction pour le film de genre Drama uniquement est nulle.

RÉSULTAT DE SORTIE:

- La sortie de l'algorithme sera de la forme suivante :

```
1,24,Powder (1995),102.88864922535025
1,32,Twelve Monkeys (a.k.a. 12 Monkeys) (1995),103.37021298654203
1,22,Copycat (1995),103.19448596542803
1,40,"Cry,101.4959765781013
1,14,Nixon (1995),101.4959765781013
1,42,Dead Presidents (1995),100.82744563938586
1,43,Restoration (1995),101.4959765781013
1,46,How to Make an American Quilt (1995),102.0386915326735
1,45,To Die For (1995),102.37991037689726
1,17,Sense and Sensibility (1995),102.0386915326735
1,6,Heat (1995),101.65558299085761
1,11,"American President,101.82715668873493
```

APPROCHE TECHNIQUE

DEMARCHE SUIVIE :

1. Générer des scores de genre normalisés pour les films
   - Chaque genre est divisé par sqrt(total des genres)
2. Générer 1 ou -1 en fonction de l'évaluation de l'utilisateur
   - >=2,5 signifie +1
   - <2,5 signifie -1
3. Attacher le +1 ou -1 à l'étape1
4. Multiplication de +1 ou -1 avec les scores de genre de film pour un utilisateur
5. Générer les scores IDF à l'aide du fichier movies.csv
6. Obtenir les scores de genre utilisateur dans une ligne
   - L'entrée est la sortie de l'étape 4
7. Multiplier les scores IDF de genre avec les scores de genre utilisateur et les associer aux scores de genre de film
8. Éliminer les films déjà vus par l'utilisateur de la recommandation finale
9. Multiplier à la fois les partitions de genre basées sur l'utilisateur et sur le film et de les trier par ordre décroissant en fonction de la partition

JOB 1 :

- Ce job a pour but de générer des scores de genre normalisés pour les films (chaque genre est divisé par sqrt(total des genres))
- L'entrée de Mapper est Movies.csv :

Mapper 1 :

```
public static class Job1Mapper extends Mapper<LongWritable, Text, LongWritable,Text>{

    @Override
    protected void map(LongWritable key, Text value,Mapper<LongWritable, Text,LongWritable, Text>.Context context)
            throws IOException, InterruptedException {
        String[] split = value.toString().split(",");
        String[] genres = {"Action","Adventure","Animation","Children","Comedy","Crime","Documentary","Drama","Fantasy"
        //k2 contains the movieid
        long k2 = Long.parseLong(split[0]);

        int length = split.length -1;
        String[] genresplit = split[length].split("\\|");
        String append="";

        for(int i=0; i<genres.length; i++){
            double score = 0.0;
            for (int j = 0; j < genresplit.length; j++) {
                if(genres[i].equals(genresplit[j]))
                {
                    score=(double)1/Math.sqrt(genresplit.length);
                }
            }
            append = genres[i]+":"+score+","+append;;
        }
        context.write(new LongWritable(k2), new Text(append));
    }
}
```

Reducer 1 :

```
public static class Job1Reducer extends Reducer<LongWritable, Text, LongWritable, Text>{
    @Override
    protected void reduce(LongWritable k2, Iterable<Text> v2s,
            Reducer<LongWritable, Text, LongWritable, Text>.Context context)
            throws IOException, InterruptedException {
        String result = "";
        for (Text genre : v2s) {
            result=result+genre;
        }
        context.write(k2, new Text(result));
    }
}
```

Driver 1 :

```java
Configuration conf1 = new Configuration();
@SuppressWarnings("deprecation")
Job job1 = new Job(conf1, "step1");
job1.setJarByClass(FinalContent.class);
FileInputFormat.setInputPaths(job1, new Path(args[0]));//input is movies.csv
job1.setInputFormatClass(TextInputFormat.class);
job1.setMapperClass(Job1Mapper.class);
job1.setMapOutputKeyClass(LongWritable.class);
job1.setMapOutputValueClass(Text.class);
job1.setReducerClass(Job1Reducer.class);
FileOutputFormat.setOutputPath(job1, new Path(output+1));
job1.setOutputFormatClass(TextOutputFormat.class);
job1.setOutputKeyClass(LongWritable.class);
job1.setOutputValueClass(Text.class);
job1.waitForCompletion(true);
```

- ✓ Le résultat correspond aux genres de films normalisés (movieid genre: Score)

Résultat :

```
1	(no genres listed):0.0,Western:0.0,War:0.0,Thriller:0.0,Sci-Fi:0.0,Romance:0.0,Mystery:0.0,Musical:0.0,Horror:0.0,Film-Noir:0.0,Fant
2	(no genres listed):0.0,Western:0.0,War:0.0,Thriller:0.0,Sci-Fi:0.0,Romance:0.0,Mystery:0.0,Musical:0.0,Horror:0.0,Film-Noir:0.0,Fant
3	(no genres listed):0.0,Western:0.0,War:0.0,Thriller:0.0,Sci-Fi:0.0,Romance:0.7071067811865475,Mystery:0.0,Musical:0.0,Horror:0.0,Fil
4	(no genres listed):0.0,Western:0.0,War:0.0,Thriller:0.0,Sci-Fi:0.0,Romance:0.5773502691896258,Mystery:0.0,Musical:0.0,Horror:0.0,Fil
5	(no genres listed):0.0,Western:0.0,War:0.0,Thriller:0.0,Sci-Fi:0.0,Romance:0.0,Mystery:0.0,Musical:0.0,Horror:0.0,Film-Noir:0.0,Fant
6	(no genres listed):0.0,Western:0.0,War:0.0,Thriller:0.5773502691896258,Sci-Fi:0.0,Romance:0.0,Mystery:0.0,Musical:0.0,Horror:0.0,Fil
```

JOB 2

- ➢ Job pour générer (1 ou -1) en fonction de l'évaluation de l'utilisateur (>=2,5 signifie +1 et <2,5 signifie -1)

Mapper 2 :

- ➢ Entrée sera rating.csv

```java
public static class Job2Mapper extends Mapper<LongWritable, Text, Text, IntWritable>{
    @Override
    protected void map(LongWritable key, Text value,Context context)
            throws IOException, InterruptedException {
        String[] linesSplit = value.toString().split(",");
        String userId = linesSplit[0];
        int userRating = -1;
        if(Double.parseDouble(linesSplit[2])>=2.5){
            userRating=1;
        }
        String userMovie= userId+":"+linesSplit[1];
        System.out.println(userMovie+" and "+userRating);
        context.write(new Text(userMovie),new IntWritable(userRating) );
        }
}
```

Reducer 2 :

```java
public static class Job2Reducer extends Reducer<Text, IntWritable, Text, IntWritable>{
    @Override
    protected void reduce(Text k2, Iterable<IntWritable> v2s,
            Reducer<Text, IntWritable, Text, IntWritable>.Context context)
            throws IOException, InterruptedException {
        System.out.println(x: "reached reducer");
        int count=0;
        for (IntWritable num : v2s) {
            count += num.get();
        }
        context.write(k2, new IntWritable(count));
    }
}
```

Driver 2 :

```java
Configuration conf2 = new Configuration();
@SuppressWarnings("deprecation")
Job job2 = new Job(conf2, "step2");
job2.setJarByClass(FinalContent.class);
FileInputFormat.setInputPaths(job2, new Path(args[1]));//input is ratings.csv
job2.setMapperClass(Job2Mapper.class);
job2.setReducerClass(Job2Reducer.class);
job2.setOutputFormatClass(TextOutputFormat.class);
job2.setOutputKeyClass(Text.class);
job2.setOutputValueClass(IntWritable.class);
FileOutputFormat.setOutputPath(job2, new Path(output+2));//output is user movie 1 or -1
job2.waitForCompletion(true);
```

Résultat :

```
87:780   1
87:785   1
87:786   1
87:802   1
87:852   -1
87:88    1
87:95    1
88:1028  1
88:104   -1
88:1073  1
88:1079  1
88:1080  1
88:1097  1
88:11    1
88:1101  1
88:1136  1
88:1183  1
88:1193  1
88:1196  1
88:1197  1
88:1207  1
88:1225  1
88:1246  1
88:1258  1
88:1259  1
88:1265  1
88:1270  1
88:1278  1
88:1282  1
```

JOB 3

Mapper 3 :

- L'entrée du mapper 1 sera la sortie de la dernière tache :

```
public static class Job3Mapper1 extends
        Mapper<LongWritable, Text, Text, Text> {
    @Override
    protected void map(LongWritable key, Text value,
            Mapper<LongWritable, Text, Text, Text>.Context context)
            throws IOException, InterruptedException {
        String[] split = value.toString().split("\t");
        context.write(new Text(split[0]), new Text(split[1]));
    }
}
```

- L'entrée du mapper 2 sera la sortie de la 1ère tache :

```java
public static class Job3Mapper2 extends
        Mapper<LongWritable, Text, Text, Text> {
    @Override
    protected void map(LongWritable key, Text value,
            Mapper<LongWritable, Text, Text, Text>.Context context)
            throws IOException, InterruptedException {
        String[] split = value.toString().split("\t");
        for (int i = 1; i <= 50; i++) {
            String k2 = i + ":" + split[0];
            context.write(new Text(k2), new Text(split[1]));
        }
    }
}
```

Reducer 3 :

- Reducer pour ajouter +1 ou -1 aux scores des genres

```java
public static class Job3Reducer extends
        Reducer<Text, Text, Text, Text> {
    @Override
    protected void reduce(Text k2, Iterable<Text> v2s,
            Reducer<Text, Text, Text, Text>.Context context)
            throws IOException, InterruptedException {
        long count = 0L;
        String result = "";
        String rating = "";
        String genre = "";
        for (Text single : v2s) {
            count++;
            if(single.toString().equals("1")||single.toString().equals("-1"))
                rating = "###" + single;
            else
                {
                    genre = single.toString();
                }
            result = genre + rating;

            }
            if (count > 1) {
                context.write(k2, new Text(result));
            }
        }
}
```

Driver 3 :

```
Configuration conf3 = new Configuration();
@SuppressWarnings("deprecation")
Job job3 = new Job(conf3, "step3");
job3.setJarByClass(FinalContent.class);
MultipleInputs.addInputPath(job3, new Path(output+2), TextInputFormat.class, Job3Mapper1.class);
MultipleInputs.addInputPath(job3, new Path(output+1), TextInputFormat.class, Job3Mapper2.class);
job3.setReducerClass(Job3Reducer.class);
job3.setOutputFormatClass(TextOutputFormat.class);
job3.setOutputKeyClass(Text.class);
job3.setOutputValueClass(Text.class);
FileOutputFormat.setOutputPath(job3, new Path(output+3));
job3.waitForCompletion(true);
```

Résultat :

```
10:1036 (no genres listed):0.0,Western:0.0,War:0.0,Thriller:0.5773502691896258,Sci-Fi:0.0,Romance:0.0,Mystery:0.0,Musical:0.0,
10:1089 (no genres listed):0.0,Western:0.0,War:0.0,Thriller:0.5773502691896258,Sci-Fi:0.0,Romance:0.0,Mystery:0.57735026918962
10:1101 (no genres listed):0.0,Western:0.0,War:0.0,Thriller:0.0,Sci-Fi:0.0,Romance:0.7071067811865475,Mystery:0.0,Musical:0.0,
10:1127 (no genres listed):0.0,Western:0.0,War:0.0,Thriller:0.5,Sci-Fi:0.5,Romance:0.0,Mystery:0.0,Musical:0.0,Horror:0.0,Film
10:1196 (no genres listed):0.0,Western:0.0,War:0.0,Thriller:0.0,Sci-Fi:0.5773502691896258,Romance:0.0,Mystery:0.0,Musical:0.0,
10:1197 (no genres listed):0.0,Western:0.0,War:0.0,Thriller:0.0,Sci-Fi:0.0,Romance:0.4472135954999579,Mystery:0.0,Musical:0.0,
10:1198 (no genres listed):0.0,Western:0.0,War:0.0,Thriller:0.0,Sci-Fi:0.0,Romance:0.0,Mystery:0.0,Musical:0.0,Horror:0.0,Film
10:1200 (no genres listed):0.0,Western:0.0,War:0.0,Thriller:0.0,Sci-Fi:0.5,Romance:0.0,Mystery:0.0,Musical:0.0,Horror:0.5,Film
10:1210 (no genres listed):0.0,Western:0.0,War:0.0,Thriller:0.0,Sci-Fi:0.5773502691896258,Romance:0.0,Mystery:0.0,Musical:0.0,
10:1220 (no genres listed):0.0,Western:0.0,War:0.0,Thriller:0.0,Sci-Fi:0.0,Romance:0.0,Mystery:0.0,Musical:0.5773502691896258,
10:1240 (no genres listed):0.0,Western:0.0,War:0.0,Thriller:0.5773502691896258,Sci-Fi:0.5773502691896258,Romance:0.0,Mystery:0
10:1291 (no genres listed):0.0,Western:0.0,War:0.0,Thriller:0.0,Sci-Fi:0.0,Romance:0.0,Mystery:0.0,Musical:0.0,Horror:0.0,Film
```

SUITE :

```
0,Drama:0.0,Documentary:0.0,Crime:0.5773502691896258,Comedy:0.0,Children:0.0,Animation:0.0,Adventure:0.0,Action:0.5773502691896258,###1
:0.0,Fantasy:0.0,Drama:0.0,Documentary:0.0,Crime:0.5773502691896258,Comedy:0.0,Children:0.0,Animation:0.0,Adventure:0.0,Action:0.0,###1
0,Drama:0.0,Documentary:0.0,Crime:0.0,Comedy:0.0,Children:0.0,Animation:0.0,Adventure:0.0,Action:0.7071067811865475,###-1
umentary:0.0,Crime:0.0,Comedy:0.0,Children:0.0,Animation:0.0,Adventure:0.5,Action:0.5,###1
0,Drama:0.0,Documentary:0.0,Crime:0.0,Comedy:0.0,Children:0.0,Animation:0.0,Adventure:0.5773502691896258,Action:0.5773502691896258,###1
4472135954999579,Drama:0.0,Documentary:0.0,Crime:0.0,Comedy:0.4472135954999579,Children:0.0,Animation:0.0,Adventure:0.4472135954999579,Action
umentary:0.0,Crime:0.0,Comedy:0.0,Children:0.0,Animation:0.0,Adventure:0.7071067811865475,Action:0.7071067811865475,###1
umentary:0.0,Crime:0.0,Comedy:0.0,Children:0.0,Animation:0.0,Adventure:0.5,Action:0.5,###1
0,Drama:0.0,Documentary:0.0,Crime:0.0,Comedy:0.0,Children:0.0,Animation:0.0,Adventure:0.5773502691896258,Action:0.5773502691896258,###1
0,Drama:0.0,Documentary:0.0,Crime:0.0,Comedy:0.5773502691896258,Children:0.0,Animation:0.0,Adventure:0.0,Action:0.5773502691896258,###1
:0.0,Fantasy:0.0,Drama:0.0,Documentary:0.0,Crime:0.0,Comedy:0.0,Children:0.0,Animation:0.0,Adventure:0.0,Action:0.5773502691896258,###1
```

JOB 4 :

Mapper 4:

```java
public static class Job4Mapper extends Mapper<LongWritable, Text, Text, DoubleWritable>{
    @Override
    protected void map(LongWritable key, Text value,
            Mapper<LongWritable, Text, Text, DoubleWritable>.Context context)
            throws IOException, InterruptedException {
        String[] split = value.toString().split("\t");
        String[] usermoviesplit = split[0].split(regex: ":");
        String user=usermoviesplit[0];
        String movie=usermoviesplit[1];
        String[] genresrating = split[1].split(regex: "###");
        String genres = genresrating[0];
        int rating = Integer.parseInt(genresrating[1]);
        String[] genresplit=genres.trim().split(regex: ",");

        for (int j = 0; j < genresplit.length; j++) {
            String[] genrescore = genresplit[j].split(regex: ":");
            String genre = genrescore[0];
            double score = Double.parseDouble(genrescore[1]);
            score = score*rating;
            String k = user+":"+genre;

            context.write(new Text(k), new DoubleWritable(score));
        }
    }
}
```

Reducer 4:

```java
public static class Job4Reducer extends Reducer<Text, DoubleWritable, Text, DoubleWritable>{
    @Override
    protected void reduce(Text k2, Iterable<DoubleWritable> v2s,
            Reducer<Text, DoubleWritable, Text, DoubleWritable>.Context context)
            throws IOException, InterruptedException {
        Double count = 0.0;
        for (DoubleWritable num : v2s) {
            count += num.get();
        }
        context.write(k2, new DoubleWritable(count));
    }
}
```

Driver 4:

```java
Configuration conf4 = new Configuration();
@SuppressWarnings("deprecation")
Job job4 = new Job(conf4, "step4");
job4.setJarByClass(FinalContent.class);
FileInputFormat.setInputPaths(job4, new Path(output+3));
job4.setMapperClass(Job4Mapper.class);
job4.setReducerClass(Job4Reducer.class);
job4.setOutputKeyClass(Text.class);
job4.setOutputValueClass(DoubleWritable.class);
FileOutputFormat.setOutputPath(job4, new Path(output+4));
job4.waitForCompletion(true);
```

Résultat :

```
10:(no genres listed)   0.0
10:Action       9.167929882896177
10:Adventure    6.040691560941889
10:Animation    0.0
10:Children     0.0
10:Comedy       5.2228543157461065
10:Crime        4.223614639131598
10:Documentary  0.0
10:Drama        9.396961063805792
10:Fantasy      1.0245638646895838
10:Film-Noir    0.0
10:Horror       3.2320508075688776
10:Musical      0.5773502691896258
10:Mystery      2.4391575887554247
10:Romance      3.0685339390596003
10:Sci-Fi       4.464101615137755
10:Thriller     8.403259203893182
```

JOB 5 :

Mapper 5:

```java
public static class Job5Mapper extends Mapper<LongWritable, Text, Text, LongWritable>{

    @Override
    protected void map(LongWritable key, Text value,
            Mapper<LongWritable, Text,Text, LongWritable>.Context context)
            throws IOException, InterruptedException {
        String[] split = value.toString().split(",");
        int length = split.length -1;
        String[] genresplit = split[length].split(regex: "\\|");

        for (int j = 0; j < genresplit.length; j++) {
            context.write(new Text(genresplit[j]), new LongWritable(1L));
        }
    }
}
```

Reducer 5:

```java
public static class Job5Reducer extends Reducer<Text, LongWritable, Text, DoubleWritable>{
    @Override
    protected void reduce(Text k2, Iterable<LongWritable> v2s,
            Reducer<Text, LongWritable, Text, DoubleWritable>.Context context)
            throws IOException, InterruptedException {
        Long count = 0L;
        for (LongWritable num : v2s) {
            count += num.get();
        }
        double finalcount = Math.log(40000/count);//total number of movies in movies.csv
        context.write(k2, new DoubleWritable(finalcount));
    }
}
```

Driver 5:

```java
Configuration conf5 = new Configuration();
@SuppressWarnings("deprecation")
Job job5 = new Job(conf5, "step5");
job5.setJarByClass(FinalContent.class);
FileInputFormat.setInputPaths(job5, new Path(args[0]));
job5.setInputFormatClass(TextInputFormat.class);
job5.setMapperClass(Job5Mapper.class);
job5.setMapOutputKeyClass(Text.class);
job5.setMapOutputValueClass(LongWritable.class);
job5.setReducerClass(Job5Reducer.class);
job5.setOutputKeyClass(Text.class);
job5.setOutputValueClass(DoubleWritable.class);
FileOutputFormat.setOutputPath(job5, new Path(output+5));
job5.setOutputFormatClass(TextOutputFormat.class);
job5.waitForCompletion(true);
```

Résultat :

```
Children        3.044522437723423
Comedy  1.0986122886681098
Crime   2.302585092994046
Documentary     2.3978952727983707
Drama   0.6931471805599453
Fantasy 2.995732273553991
Film-Noir       4.736198448394496
Horror  2.302585092994046
IMAX    5.313205979041787
Musical 3.6109179126442243
Mystery 2.9444389791664403
Romance 1.9459101490553132
Sci-Fi  2.772588722239781
Thriller        1.791759469228055
War     3.295836866004329
```

JOB 6 :

Mapper 6:

```java
public static class Job6Mapper extends Mapper<LongWritable, Text, LongWritable, Text>{

    @Override
    protected void map(LongWritable key, Text value,
            Mapper<LongWritable, Text, LongWritable, Text>.Context context)
            throws IOException, InterruptedException {
        String[] split = value.toString().split("\t");
        //System.out.println(split[0]+split[1]);
        String[] usergenre = split[0].split(regex: ":");
        double score = Double.parseDouble(split[1]);
        long k2 = Long.parseLong(usergenre[0]);
        String tmp = usergenre[1]+":"+score;
        context.write(new LongWritable(k2), new Text(tmp));
    }
}
```

Reducer 6:

```java
public static class Job6Reducer extends Reducer<LongWritable, Text, LongWritable, Text>{
    @Override
    protected void reduce(LongWritable k2, Iterable<Text> v2s,
            Reducer<LongWritable, Text, LongWritable, Text>.Context context)
            throws IOException, InterruptedException {
        String v3 = "";
        for (Text v2 : v2s) {
            //combine single users all movie ratings
            v3 = v3+","+v2.toString();
            //System.out.println(v2.toString());
        }
        context.write(k2, new Text(v3.substring(beginIndex: 1)));
    }
}
```

Driver 6:

```java
Configuration conf6 = new Configuration();
@SuppressWarnings("deprecation")
Job job6 = new Job(conf6, "step6");
job6.setJarByClass(FinalContent.class);
FileInputFormat.setInputPaths(job6, new Path(output+4));
job6.setInputFormatClass(TextInputFormat.class);
job6.setMapperClass(Job6Mapper.class);
job6.setMapOutputKeyClass(LongWritable.class);
job6.setMapOutputValueClass(Text.class);
job6.setReducerClass(Job6Reducer.class);
FileOutputFormat.setOutputPath(job6, new Path(output+6));
job6.setOutputFormatClass(TextOutputFormat.class);
job6.waitForCompletion(true);
```

Résultat :

```
Action:-0.5,(no genres listed):0.0,Western:0.7071067811865475,War:-0.7071067811865475,Thriller:1.0773502691896257,Sci-Fi:0.7844570503761732,Romance:0.5,Myster
(no genres listed):0.0,Western:1.0773502691896257,War:2.861807319565799,Thriller:9.980229311390062,Sci-Fi:2.861807319565799,Romance:13.689489803689739,Mystery
War:3.776020881938894,Western:0.5773502691896258,(no genres listed):0.0,Action:8.566015749016966,Adventure:5.126477998568794,Animation:1.2460720862226422,Chil
(no genres listed):0.0,Action:30.860059151552424,Adventure:33.031882775895404,Animation:10.933215789917849,Children:20.957415175677923,Comedy:49.2221645137219
Western:0.0,War:2.4915638315627207,Thriller:5.886371184255383,Sci-Fi:3.6547005383792515,Romance:22.842105037178673,Mystery:2.439157588755425,Musical:5.0972949
Western:0.0,War:1.6019141338792096,Thriller:0.2401068143134103,Sci-Fi:0.9717774601895419,Romance:0.5773502691896258,Mystery:0.4472135954999579,Musical:0.0,Hor
Western:0.07735026918962584,Action:11.646052776687698,Adventure:15.218403706488393,Animation:5.7072650968067675,Children:7.12185882087261,Comedy:24.0806384812
War:3.361807319565799,Thriller:19.659129235529388,Sci-Fi:7.427823068582765,Romance:12.929976779695895,Mystery:7.350092637700392,Musical:0.5773502691896258,Hor
Western:0.0,War:0.7071067811865474,Thriller:4.723614639131599,Sci-Fi:2.439157588755425,Romance:9.250712376627058,Mystery:1.2844570503761732,Musical:0.44721359
(no genres listed):0.0,Western:-0.5773502691896258,War:0.4472135954999579,Thriller:8.403259203893182,Sci-Fi:4.464101615137755,Romance:3.0685339390596003,Myste
Action:5.50444247157802,(no genres listed):0.0,Western:1.0773502691896257,War:1.2071067811865475,Thriller:6.659143009957273,Sci-Fi:1.902528337698811,Romance:1
Western:0.0,War:0.0,Thriller:3.7236146391315983,Sci-Fi:0.867772591475031,Romance:-0.9138334006803486,Mystery:1.8618073195657991,Musical:-0.3698633263103321,Ho
Western:1.7844570503761732,War:2.8618073195657994,Thriller:4.309020915065757,Sci-Fi:3.1019141338792093,Romance:5.371969743908872,Mystery:2.1927053408400363,Mu
```

JOB 7:

Mapper 7:

```java
public static class Job7Mapper1 extends
        Mapper<LongWritable, Text, Text, Text> {

    @Override
    protected void map(LongWritable key, Text value,
            Mapper<LongWritable, Text, Text, Text>.Context context)
            throws IOException, InterruptedException {

        HashMap<String,Double> hm = new HashMap<String,Double>();
        hm.put(key: "(no genres listed)", new Double(value: 2.9444389791664403));
        hm.put(key: "Action",new Double(value: 1.9459101490553132));
        hm.put(key: "Adventure",new Double(value: 2.5649493574615367));
        hm.put(key: "Animation",new Double(value: 3.1780538303479458));
        hm.put(key: "Children",new Double(value: 3.044522437723423));
        hm.put(key: "Comedy",new Double(value: 1.0986122886681098));
        hm.put(key: "Crime",new Double(value: 2.302585092994046));
        hm.put(key: "Documentary",new Double(value: 2.3978952727983707));
        hm.put(key: "Drama",new Double(value: 0.6931471805599453));
        hm.put(key: "Fantasy",new Double(value: 2.995732273553991));
        hm.put(key: "Film-Noir",new Double(value: 4.736198448394496));
        hm.put(key: "Horror",new Double(value: 2.302585092994046));
        hm.put(key: "IMAX",new Double(value: 5.313205979041787));
        hm.put(key: "Musical",new Double(value: 3.6109179126442243));
        hm.put(key: "Mystery",new Double(value: 2.9444389791664403));
        hm.put(key: "Romance",new Double(value: 1.9459101490553132));
        hm.put(key: "Sci-Fi",new Double(value: 2.772588722239781));
        hm.put(key: "Thriller",new Double(value: 1.791759469228055));
        hm.put(key: "War",new Double(value: 3.295836866004329));
        hm.put(key: "Western",new Double(value: 3.713572066704308));
```

```
        String[] split = value.toString().split("[\t,]");//movie and genre:score
        for (int i = 1; i <= 50; i++) {//total movies
            String v3 = "";
            String k2="";
            String v2 = "";
            for (int j = 1; j < split.length; j++) {
                String[] genrescore = split[j].split(regex: ":");//splits genre and score
                double score = Double.parseDouble(genrescore[1]);
                double hashscore = hm.get(genrescore[0]);
                score = score*hashscore;
                k2 =  split[0]+":"+i;//userid:movieid
                v2 = genrescore[0]+":"+score;
                v3 = v3+","+v2.toString();
            }
            context.write(new Text(k2), new Text(v3.substring(beginIndex: 1)));//userid:movieid genre:rating
        }
    }
}
```

Reducer 7:

```
public static class Job7Reducer extends
        Reducer<Text, Text, Text, Text> {
    @Override
    protected void reduce(Text k2, Iterable<Text> v2s,
            Reducer<Text, Text, Text, Text>.Context context)
            throws IOException, InterruptedException {
        long count = 0L;
        double result = 1;
        String s = "";
        for (Text single : v2s) {
            count++;
            s = s+","+single;
            if (count > 1) {
                context.write(k2, new Text(s.substring(beginIndex: 1)));//movieid:movieid:userid
            }
        }
    }
}
```

Driver 7:

```
Configuration conf7 = new Configuration();
@SuppressWarnings("deprecation")
Job job7 = new Job(conf7, "step7");
job7.setJarByClass(FinalContent.class);
MultipleInputs.addInputPath(job7, new Path(output+6), TextInputFormat.class, Job7Mapper1.class);
MultipleInputs.addInputPath(job7, new Path(output+1), TextInputFormat.class, Job7Mapper2.class);
job7.setReducerClass(Job7Reducer.class);
job7.setOutputFormatClass(TextOutputFormat.class);
job7.setOutputKeyClass(Text.class);
job7.setOutputValueClass(Text.class);
FileOutputFormat.setOutputPath(job7, new Path(output+7));
job7.waitForCompletion(true);
```

Résultat :

```
10:1    (no genres listed):0.0,Western:-2.1440318323668075,War:1.473943055027109,Thriller:15.056619250953414,Sci-Fi:12.377117793063332,Romance:5.971091334736754
10:10   (no genres listed):0.0,Western:0.0,War:0.0,Thriller:0.5773502691896258,Sci-Fi:0.0,Romance:0.0,Mystery:0.0,Musical:0.0,Horror:0.0,Film-Noir:0.0,Fantasy:0
10:11   (no genres listed):0.0,Western:0.0,War:0.0,Thriller:0.0,Sci-Fi:0.0,Romance:0.5773502691896258,Mystery:0.0,Musical:0.0,Horror:0.0,Film-Noir:0.0,Fantasy:0
10:12   (no genres listed):0.0,Western:0.0,War:0.0,Thriller:0.0,Sci-Fi:0.0,Romance:0.0,Mystery:0.0,Musical:0.0,Horror:0.7071067811865475,Film-Noir:0.0,Fantasy:0
10:13   (no genres listed):0.0,Western:0.0,War:0.0,Thriller:0.0,Sci-Fi:0.0,Romance:0.0,Mystery:0.0,Musical:0.0,Horror:0.0,Film-Noir:0.0,Fantasy:0.0,Drama:0.0,Do
10:14   (no genres listed):0.0,Western:0.0,War:0.0,Thriller:0.0,Sci-Fi:0.0,Romance:0.0,Mystery:0.0,Musical:0.0,Horror:0.0,Film-Noir:0.0,Fantasy:0.0,Drama:1.0,Do
10:15   (no genres listed):0.0,Western:0.0,War:0.0,Thriller:0.0,Sci-Fi:0.0,Romance:0.5773502691896258,Mystery:0.0,Musical:0.0,Horror:0.0,Film-Noir:0.0,Fantasy:0
10:16   (no genres listed):0.0,Western:0.0,War:0.0,Thriller:0.0,Sci-Fi:0.0,Romance:0.0,Mystery:0.0,Musical:0.0,Horror:0.0,Film-Noir:0.0,Fantasy:0.0,Drama:0.7071
10:17   (no genres listed):0.0,Western:0.0,War:0.0,Thriller:0.0,Sci-Fi:0.0,Romance:0.7071067811865475,Mystery:0.0,Musical:0.0,Horror:0.0,Film-Noir:0.0,Fantasy:0
10:18   (no genres listed):0.0,Western:0.0,War:0.0,Thriller:0.0,Sci-Fi:0.0,Romance:0.0,Mystery:0.0,Musical:0.0,Horror:0.0,Film-Noir:0.0,Fantasy:0.0,Drama:0.0,Do
10:19   (no genres listed):0.0,Western:0.0,War:0.0,Thriller:0.0,Sci-Fi:0.0,Romance:0.0,Mystery:0.0,Musical:0.0,Horror:0.0,Film-Noir:0.0,Fantasy:0.0,Drama:0.0,Do
10:2    (no genres listed):0.0,Western:0.0,War:0.0,Thriller:0.0,Sci-Fi:0.0,Romance:0.0,Mystery:0.0,Musical:0.0,Horror:0.0,Film-Noir:0.0,Fantasy:0.57735026918962
10:20   (no genres listed):0.0,Western:-2.1440318323668075,War:1.473943055027109,Thriller:15.056619250953414,Sci-Fi:12.377117793063332,Romance:5.971091334736754
```

JOB 8 :

Mapper 8:

```java
public static class Job9Mapper1 extends Mapper<LongWritable, Text, Comparator1, Text> {
    @Override
    protected void map(LongWritable key,Text value,
            Mapper<LongWritable, Text, Comparator1, Text>.Context context)
            throws IOException, InterruptedException {
        //value is userid   movieid:rating etc...   1   1953:4.0,2150:3.0....etc
        String[] split = value.toString().split("[,]"); //split -> [1,1953:4.0,2150:3.0....]
        context.write(new Comparator1(Long.parseLong(split[0]), Long.parseLong(split[1])), new Text(split[2]));

    }
}
```

```java
public static class Job9Mapper2 extends Mapper<LongWritable, Text, Comparator1, Text> {
    @Override
    protected void map(LongWritable key,Text value,
            Mapper<LongWritable, Text, Comparator1, Text>.Context context)
            throws IOException, InterruptedException {
        // userid   movieid,sum of the scores userid:movieid pairs
        //value is 1(userid)    1(movieid),12.0 (sum of the scores userid:movieid pairs)
        String[] split = value.toString().split("[\t]"); //split -> [1,1,12.0] (userid,movieid,sum of scores)
        String[] split2 = split[0].split(regex: ":");
        context.write(new Comparator1(Long.parseLong(split2[0]), Long.parseLong(split2[1])), new Text(split[1]));

    }
}
```

Reducer 8:

```
public static class Job9Reducer extends Reducer<Comparator1, Text, Text, Text> {
    @Override
    protected void reduce(Comparator1 k2,Iterable<Text> v2s,
            Reducer<Comparator1, Text, Text, Text>.Context context)
            throws IOException, InterruptedException {
        ArrayList<String> list = new ArrayList<String>();
        for (Text time : v2s) {
            list.add(time.toString());
        }
        if ((list.size() == 1)&&list.get(index: 0).length()>20)  {
            String finalkey=k2.first+":"+k2.second;
            String v3 = list.get(index: 0);
            context.write(new Text(finalkey), new Text(v3));//

        }
    }
}
```

Driver 8:

```
Configuration conf9 = new Configuration();
@SuppressWarnings("deprecation")
Job job9 = new Job(conf9,"step9");
job9.setJarByClass(FinalContent.class);
MultipleInputs.addInputPath(job9, new Path(args[1]), TextInputFormat.class, Job9Mapper1.class);
MultipleInputs.addInputPath(job9, new Path(output+7), TextInputFormat.class, Job9Mapper2.class);
job9.setMapOutputKeyClass(Comparator1.class);
job9.setMapOutputValueClass(Text.class);
job9.setReducerClass(Job9Reducer.class);
job9.setOutputKeyClass(Text.class);
job9.setOutputValueClass(Text.class);
FileOutputFormat.setOutputPath(job9, new Path(output+8));
job9.waitForCompletion(true);
```

Résultat:

```
1:1     (no genres listed):0.0,Western:0.0,War:0.0,Thriller:0.0,Sci-Fi:0.0,Romance:0.0,Mystery:0.0,Musical:0.0,Horror:0.0,Film-Noir:0.0,Fantasy:0.44721359549
1:2     (no genres listed):0.0,Western:0.0,War:0.0,Thriller:0.0,Sci-Fi:0.0,Romance:0.0,Mystery:0.0,Musical:0.0,Horror:0.0,Film-Noir:0.0,Fantasy:0.57735026918
1:3     (no genres listed):0.0,Western:0.0,War:0.0,Thriller:0.0,Sci-Fi:0.0,Romance:0.7071067811865475,Mystery:0.0,Musical:0.0,Horror:0.0,Film-Noir:0.0,Fantas
1:4     (no genres listed):0.0,Western:0.0,War:0.0,Thriller:0.0,Sci-Fi:0.0,Romance:0.5773502691896258,Mystery:0.0,Musical:0.0,Horror:0.0,Film-Noir:0.0,Fantas
1:5     (no genres listed):0.0,Western:0.0,War:0.0,Thriller:0.0,Sci-Fi:0.0,Romance:0.0,Mystery:0.0,Musical:0.0,Horror:0.0,Film-Noir:0.0,Fantasy:0.0,Drama:0.0
1:6     (no genres listed):0.0,Western:0.0,War:0.0,Thriller:0.5773502691896258,Sci-Fi:0.0,Romance:0.0,Mystery:0.0,Musical:0.0,Horror:0.0,Film-Noir:0.0,Fantas
1:7     (no genres listed):0.0,Western:0.0,War:0.0,Thriller:0.0,Sci-Fi:0.0,Romance:0.7071067811865475,Mystery:0.0,Musical:0.0,Horror:0.0,Film-Noir:0.0,Fantas
1:8     (no genres listed):0.0,Western:0.0,War:0.0,Thriller:0.0,Sci-Fi:0.0,Romance:0.0,Mystery:0.0,Musical:0.0,Horror:0.0,Film-Noir:0.0,Fantasy:0.0,Drama:0.0
1:9     (no genres listed):0.0,Western:0.0,War:0.0,Thriller:0.0,Sci-Fi:0.0,Romance:0.0,Mystery:0.0,Musical:0.0,Horror:0.0,Film-Noir:0.0,Fantasy:0.0,Drama:0.0
1:10    Action:-0.9729550745276566,(no genres listed):0.0,Western:2.625891990791558,War:-2.3305085976362796,Thriller:1.930352546495906,Sci-Fi:2.1749767709544
1:11    (no genres listed):0.0,Western:0.0,War:0.0,Thriller:0.0,Sci-Fi:0.0,Romance:0.5773502691896258,Mystery:0.0,Musical:0.0,Horror:0.0,Film-Noir:0.0,Fantas
1:12    (no genres listed):0.0,Western:0.0,War:0.0,Thriller:0.0,Sci-Fi:0.0,Romance:0.0,Mystery:0.0,Musical:0.0,Horror:0.7071067811865475,Film-Noir:0.0,Fantas
1:13    (no genres listed):0.0,Western:0.0,War:0.0,Thriller:0.0,Sci-Fi:0.0,Romance:0.0,Mystery:0.0,Musical:0.0,Horror:0.0,Film-Noir:0.0,Fantasy:0.0,Drama:0.0
1:14    (no genres listed):0.0,Western:0.0,War:0.0,Thriller:0.0,Sci-Fi:0.0,Romance:0.0,Mystery:0.0,Musical:0.0,Horror:0.0,Film-Noir:0.0,Fantasy:0.0,Drama:1.0
1:15    (no genres listed):0.0,Western:0.0,War:0.0,Thriller:0.0,Sci-Fi:0.0,Romance:0.5773502691896258,Mystery:0.0,Musical:0.0,Horror:0.0,Film-Noir:0.0,Fantas
1:16    (no genres listed):0.0,Western:0.0,War:0.0,Thriller:0.0,Sci-Fi:0.0,Romance:0.0,Mystery:0.0,Musical:0.0,Horror:0.0,Film-Noir:0.0,Fantasy:0.0,Drama:0.7
1:17    (no genres listed):0.0,Western:0.0,War:0.0,Thriller:0.0,Sci-Fi:0.0,Romance:0.7071067811865475,Mystery:0.0,Musical:0.0,Horror:0.0,Film-Noir:0.0,Fantas
```

JOB 9 :

Mapper 9:

```
public static class Job8Mapper extends
        Mapper<LongWritable, Text, Comparator2,Text> {

    @Override
    protected void map(LongWritable key, Text value,
            Mapper<LongWritable, Text, Comparator2,Text>.Context context)
            throws IOException, InterruptedException {

        HashMap<String,Double> hm = new HashMap<String,Double>();
        hm.put(key: "(no genres listed)", new Double(value: 1.0));
        hm.put(key: "Action",new Double(value: 1.0));
        hm.put(key: "Adventure",new Double(value: 1.0));
        hm.put(key: "Animation",new Double(value: 1.0));
        hm.put(key: "Children",new Double(value: 1.0));
        hm.put(key: "Comedy",new Double(value: 1.0));
        hm.put(key: "Crime",new Double(value: 1.0));
        hm.put(key: "Documentary",new Double(value: 1.0));
        hm.put(key: "Drama",new Double(value: 1.0));
        hm.put(key: "Fantasy",new Double(value: 1.0));
        hm.put(key: "Film-Noir",new Double(value: 1.0));
        hm.put(key: "Horror",new Double(value: 1.0));
        hm.put(key: "IMAX",new Double(value: 1.0));
        hm.put(key: "Musical",new Double(value: 1.0));
        hm.put(key: "Mystery",new Double(value: 1.0));
        hm.put(key: "Romance",new Double(value: 1.0));
        hm.put(key: "Sci-Fi",new Double(value: 1.0));
        hm.put(key: "Thriller",new Double(value: 1.0));
        hm.put(key: "War",new Double(value: 1.0));
        hm.put(key: "Western",new Double(value: 1.0));
```

```
        String[] split = value.toString().split("[\t,]");//user:movie and genre:score
        String[] usermoviesplit = split[0].split(regex: ":");
        String userid = usermoviesplit[0];
        String movieid = usermoviesplit[1];
        for (int j = 1; j < split.length; j++) {
            String[] genrescore = split[j].split(regex: ":");//splits genre and score
            double score = Double.parseDouble(genrescore[1]);
            double hashscore = hm.get(genrescore[0]);
            score = score*hashscore;
            hm.put(genrescore[0],new Double(score));
        }
        double finalvalue = 100.0;
        for (double d : hm.values()){
            finalvalue = finalvalue + d;
        }
        context.write(new Comparator2(Long.parseLong(userid), finalvalue), new Text(movieid));
    }
}
```

Reducer 9:

```
public static class Job8Reducer extends
        Reducer< Comparator2,Text, LongWritable, Text> {

    @Override
    protected void reduce(Comparator2 k2, Iterable<Text> v2s,
            Reducer<Comparator2, Text, LongWritable, Text>.Context context)
            throws IOException, InterruptedException {
        for (Text single : v2s) {
            String s = single.toString();
            String moviescore = s+","+k2.second;
                context.write(new LongWritable(k2.first), new Text(moviescore));
        }
    }
}
```

Driver 9:

```
Configuration conf8 = new Configuration();
@SuppressWarnings("deprecation")
Job job8 = new Job(conf8, "step8");
job8.setJarByClass(FinalContent.class);
FileInputFormat.setInputPaths(job8, new Path(output+8));
job8.setMapperClass(Job8Mapper.class);
job8.setReducerClass(Job8Reducer.class);
job8.setOutputFormatClass(TextOutputFormat.class);
job8.setOutputKeyClass(Text.class);
job8.setOutputValueClass(DoubleWritable.class);
//job8.setSortComparatorClass(IntComparator.class);
job8.setMapOutputKeyClass(Comparator2.class);
//job8.setMapOutputKeyClass(DoubleWritable.class);
job8.setMapOutputValueClass(Text.class);
job8.setNumReduceTasks(1);
FileOutputFormat.setOutputPath(job8, new Path(output+9));
return job8.waitForCompletion(true) ? 0 : 1;
```

Résultat:

```
1	12,102.602537913207
1	24,102.88864922535025
1	32,103.37021298654203
1	22,103.19448596542803
1	40,101.4959765781013
1	14,101.4959765781013
1	42,100.82744563938586
1	43,101.4959765781013
1	46,102.0386915326735
1	45,102.37991037689726
1	17,102.0386915326735
1	6,101.65558299085761
1	11,101.82715668873493
```

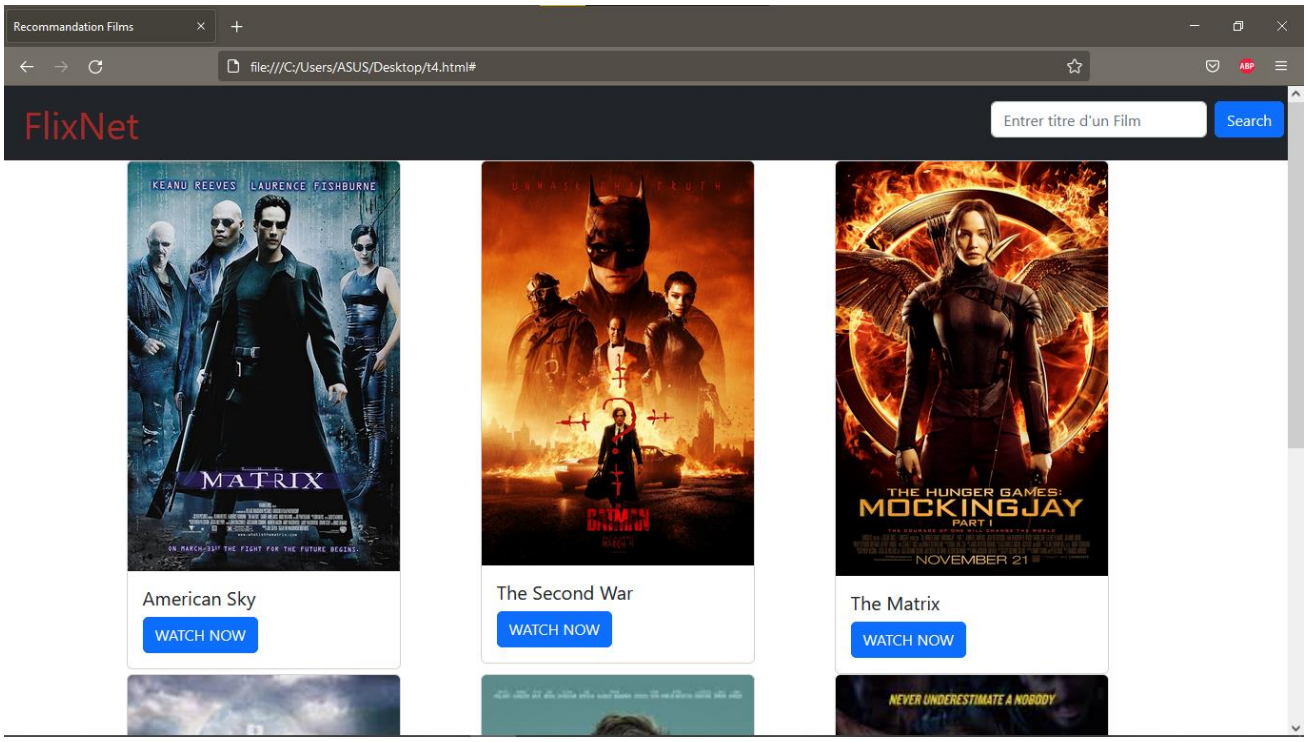
CLASSES AUXILIAIRES:

```java
public static class IntComparator extends WritableComparator {

    public IntComparator() {
        super(IntWritable.class);
    }
    @Override
    public int compare(byte[] b1, int s1, int l1, byte[] b2, int s2, int l2) {
        Integer value1 = ByteBuffer.wrap(b1, s1, l1).getInt();
        Integer value2 = ByteBuffer.wrap(b2, s2, l2).getInt();
        return value1.compareTo(value2) * (-1);
    }
}
```

```java
public static class Comparator2 implements WritableComparable<Comparator2>{
    long first;
    double second;

    public Comparator2(){}

    public Comparator2(long first, double second){
        this.first = first;
        this.second = second;
    }

    public void write(DataOutput out) throws IOException {
        out.writeLong(first);
        out.writeDouble(second);
    }

    public void readFields(DataInput in) throws IOException {
        this.first = in.readLong();
        this.second = in.readDouble();
    }

    public int compareTo(Comparator2 o) {
        int minus = (int)(this.first - o.first);
        System.out.println("minus is "+minus);
        System.out.println(this.first +" and "+ o.first);
        if(minus != 0){
            return minus;
        }
        System.out.println(this.second +" and2 "+ o.second);
        return (int)(o.second - this.second);
    }

}
```

```java
public static class Comparator1 implements WritableComparable<Comparator1> {
    long first;
    long second;

    public Comparator1() {
    }

    public Comparator1(long first, long second) {
        this.first = first;
        this.second = second;
    }

    public void write(DataOutput out) throws IOException {
        out.writeLong(first);
        out.writeLong(second);
    }

    public void readFields(DataInput in) throws IOException {
        this.first = in.readLong();
        this.second = in.readLong();
    }

    public int compareTo(Comparator1 o) {
        int minus = (int) (this.first - o.first);
        if (minus != 0) {
            return minus;
        }
        return (int) (this.second - o.second);
    }
}
```

INTERFACE WEB :



REMARQUE :

• J'ai pu créer cette interface graphique en utilisant Html et CSS, mais malheureusement je n'ai pas eu assez de temps pour le lier et fusionner avec la base de données générée par le framework Mapreduce.

• Cependant, nous nous efforcerons de poursuivre ce travail dès que possible.

CONCLUSION:

Ce mini-projet présente un système de recommandation basé sur le contenu.

Le système de recommandation est devenu de plus en plus important en raison de la surcharge d'informations. Pour le système de recommandation basé sur le contenu en particulier, nous avons essayé de trouver un moyen pour recommander un film en utilisant le Framework MapReduce.

Pour les problèmes qui s'est posé, nous utilisons d'abord un algorithme de recommandation basé sur le contenu, ce qui signifie qu'il n'y a pas des informations de démarrage pour utiliser l'algorithme pour la première fois.